Uma abordagem baseada em lógica para escalonamento *offline* de sistemas embarcados de tempo real considerando o tempo adicional do despachante

Osman Seixas Júnior e Raimundo Barreto

Departamento de Ciência da Computação – Universidade Federal do Amazonas
Manaus, AM – Brasil

{ospj,rbarreto}@dcc.ufam.edu.br

Abstract. *The offline scheduling approach requires a higher cost in the system project phase since this problem is NP-hard and the tasks' schedule must be generated in the project time. To reduce this cost, it is necessary to develop tools that support a more complex tasks set and consider the additional time spent by the dispatcher, which is often neglected by researchers. This paper presents a logic approach to build an offline schedule for embedded real time systems. We also show experiments conducted using the proposed schedule generation algorithm.*

Resumo. *A abordagem de escalonamento offline exige um maior custo na fase de projeto do sistema visto que este problema é NP-difícil e a escala das tarefas tem que ser gerada em tempo de projeto. Para amenizar este custo é necessário que sejam desenvolvidas ferramentas que dêem suporte a conjuntos de tarefas mais complexos e que considerem o tempo adicional gasto pelo despachante que é frequentemente negligenciado pelos pesquisadores. Este artigo apresenta uma abordagem lógica para construir uma escala offline para sistemas embarcados de tempo real. Também mostramos experimentos realizados utilizando o algoritmo de geração de escalas proposto.*

1. Introdução

Além de todas as atribuições de um sistema convencional, um sistema de tempo real (STR) deve cumprir o atendimento de restrições temporais que são impostas na execução de seus processos. Essa característica adicional é fundamental para que um sistema de tempo real funcione corretamente, pois, além de executarem a lógica de forma correta, os processos devem executar em um tempo hábil de modo que não comprometa o atendimento das restrições temporais. Em um STR, o não atendimento de alguma restrição temporal pode gerar conseqüências graves e irreversíveis, tais como: danos materiais, ambientais e humanos. Estes sistemas são classificados como Sistemas Críticos de Tempo Real (*Hard Real-Time Systems*) e exigem garantias de que todas as restrições temporais serão atendidas, ou seja, deve existir previsibilidade do comportamento do sistema em tempo de projeto. Neste trabalho adotaremos uma abordagem de escalonamento executada em tempo de projeto cuja escala gerada garante o atendimento das restrições impostas ao sistema, inclusive na situação de carga de pico. Desta forma podemos utilizar a ferramenta desenvolvida neste trabalho para

garantir se o conjunto de tarefas de uma dada aplicação (compatível com o modelo de tarefas descrito na Seção 4) atenderá todas as restrições durante sua execução. Esta garantia é indispensável ao desenvolvimento de aplicações de tempo-real crítico.

O restante deste trabalho está organizado como segue. A Seção 2 introduz o problema de escalonamento. Na Seção 3 são descritos os trabalhos relacionados e na Seção 4 é descrito o modelo de tarefas adotado neste trabalho. A Seção 5 apresenta a solução proposta para o problema de escalonamento. A Seção 6 apresenta a interface gráfica desenvolvida principalmente para visualização das escalas geradas na forma de Diagrama de *Gantt*. A Seção 7 discute os resultados experimentais. Finalmente, a Seção 8 apresenta as conclusões e possíveis trabalhos futuros.

2. Problema de escalonamento

Um sistema embarcado de tempo real possui um conjunto de tarefas, onde cada uma possui propriedades e relacionamentos que se apresentam na forma de restrições e devem ser consideradas durante a execução do sistema. Desta forma, a política de escalonamento adotada por um sistema de tempo real deve considerar todas as restrições impostas às tarefas. As abordagens de escalonamento de tempo real podem ser classificadas quanto ao tipo de carga do sistema (estática ou dinâmica) e ao momento em que a escala é gerada (*offline ou online*). Neste trabalho adotaremos uma abordagem de escalonamento *offline* para sistemas com carga estática. Mais especificamente, o modelo de escalonamento utilizado é o executivo cíclico [Baker and Shaw 1988].

Para considerar a carga de pico do sistema na geração da escala do sistema foi assumido que, no início da escala, todas as tarefas do sistema são instanciadas ao mesmo tempo a fim de garantir que a escala gerada pelo algoritmo atenda todas as restrições impostas, inclusive na situação de pico de carga do sistema.

3. Trabalhos relacionados

Em [Ekelin and Jonsson 1999] são descritas restrições comumente encontradas em aplicações de sistemas embarcados de tempo real. Este trabalho possibilita o desenvolvimento de algoritmos de escalonamento que dêem suporte a modelos de tarefas com relações e propriedades mais complexas. Algoritmos de escalonamento e alocação de tarefas em sistemas de tempo-real distribuídos foram elaborados por [Schild and Wurtz 1998] e [Ekelin and Jonsson 2001] e foram implementados no paradigma CLP (*Constraint Logic Programming*) que corresponde à implementação de problemas CSP (*Constraint Satisfaction Problem*) em linguagens de programação lógica. Tais trabalhos sugerem simplicidade oferecida na definição do problema de escalonamento em CSP. Entretanto, o algoritmo de busca deve ser ajustado para que a complexidade natural do problema (complexidade exponencial) seja reduzida e torne viável o tempo de execução do algoritmo para conjunto de tarefas maiores.

Este trabalho diferencia-se dos demais por fornecer um algoritmo de busca específico para o problema de escalonamento de conjunto de tarefas compatíveis com o modelo de tarefas adotado neste trabalho, contrariamente aos trabalhos anteriores que possibilitavam apenas a modelagem do problema utilizando algoritmos de busca genéricos. Além disso, o modelo de tarefas suportado pelo algoritmo considera o custo associado à ação do despachante que costuma ser negligenciado em outros trabalhos.

4. Modelo de tarefas

Um sistema de tempo real pode ser visto como um conjunto de tarefas que são executadas periodicamente durante todo o funcionamento do sistema na forma de instâncias. Em uma visão mais operacional, cada tarefa corresponde a um programa diferente e as instâncias de uma mesma tarefa são processos criados sequencialmente que estão submetidos às mesmas restrições. Baseado no trabalho proposto por [Ekelin and Jonsson 1999], as tarefas do sistema possuem as seguintes propriedades:

- Período: Duração de tempo entre a criação de dois processos consecutivos de uma mesma tarefa.
- WCET (*Worst Case Execution Time*): Tempo de duração de execução do processo no seu pior caso. Este valor deve ser estimado por ferramentas que fogem ao escopo deste trabalho.
- Liberação (*Release*): Tempo de espera a partir da criação do processo para que ele seja colocado na lista de prontos do processador.
- Prazo (*Deadline*): Tempo limite que um processo tem para finalizar sua execução. Este tempo é relativo ao tempo em que o processo foi criado.
- Preempção: Indica se a tarefa pode ter sua execução interrompida por alguma outra tarefa.

Quanto aos relacionamentos entre tarefas, temos:

- Precedência: Situação em que uma tarefa produz como saída algum dado que servirá de entrada para outra tarefa. Assim, a tarefa precedente deve terminar sua execução antes que a outra tarefa comece a executar. É importante notar que este tipo de relação exige que ambas as tarefas possuam o mesmo período de execução para que a quantidade de processos produtores e consumidores sejam iguais.
- Exclusão: Situação em que um processo, ao iniciar sua execução, proíbe que outra tarefa execute enquanto estiver executando. Neste trabalho, a relação de exclusão é definida como anti-simétrica.

Outro fator que é considerado neste modelo de tarefas é o custo causado pelas trocas de contexto ocorridas no processador. Existem dois custos associados à execução do despachante:

- Salvamento de contexto: Quando um processo é preemptado, o despachante deve salvar o contexto do processo para que futuramente seja possível retomar sua execução do ponto em que parou.
- Restauração de contexto: O despachante deve restaurar o contexto cada vez que um processo ganha o processador para iniciar ou reiniciar sua execução.

5. Algoritmo de escalonamento

Neste trabalho propomos um algoritmo de escalonamento utilizando a abordagem lógica que busca uma escala viável para o conjunto de tarefas de entrada. Os dados de

entrada necessários para execução da ferramenta correspondem ao modelo de tarefas (descrito na Seção 4) representado na Figura 1.

```
%% Tarefas e propriedades
nome_tarefa1(WCET1,Deadline1, Período1,Release1).
nome_tarefa2(WCET2,Deadline2, Período2,Release2).
nome_tarefa3(WCET3,Deadline3, Período3,Release3).
...
nome_tarefan(WCETn,Deadlinen, Períodon,Releasen).
%% onde nome_tarefai corresponde ao nome da tarefa i do conjunto.

%% Relações
relação(tarefa1,tarefa2).
relação(tarefa1,tarefa3).
...
relação(tarefa3,tarefan).
%% onde a relação pode ser: precede ou exclui.

%% Tipo Tarefa
tipo_tarefa(tarefa1).
tipo_tarefa(tarefa2).
...
tipo_tarefa(tarefan).
%% onde tipo_tarefa pode ser: preemptiva ou naoPreemptiva.

%% Custos da troca de contexto
salvar_contexto(Custo1).
restaurar_contexto(Custo2).
```

Figura 1. Representação textual do conjunto de tarefas do sistema

Com o conjunto de tarefas definido, podemos executar o algoritmo de escalonamento em busca de uma escala viável para as tarefas de modo que todas as restrições sejam atendidas. Os principais predicados que definem se as restrições do sistema são atendidas na alocação de cada slot de tempo são apresentados na Figura 2.

```
consomeWCET(Tar,Inst,Instancias).
satisfazPrecedencia(Tar,Inst,Instancias).
satisfazExclusao(Tar,Instancias).
satisfazNaoPreempcao(Tar,SlotAnt,Instancias).
atendeTrocaContexto(Despachante,(Tar,Inst),UltimoSlot,Instancias).
preveDeadlines(SlotAtual,Instancias,Tar,Inst).
```

Figura 2. Principais predicados que representam as restrições do sistema.

O algoritmo consiste em dividir todas as instâncias do sistema em **unidades de execução** que serão alocados nos *slots* de tempo disponíveis da escala a ser gerada, de modo que as restrições do sistema não sejam violadas como ilustrado na Figura 3.

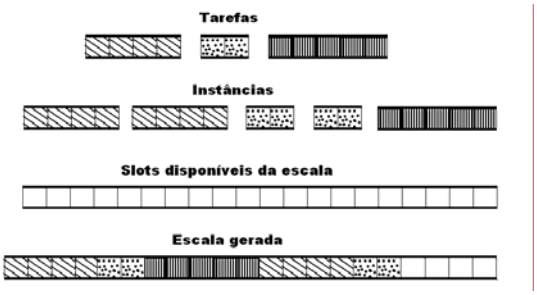

Figura 3. Visualização geral do algoritmo de escalonamento

5.1. Espaço de busca

Uma solução trivial que utiliza força-bruta seria analisar todas as combinações entre as unidades de execução das instâncias e os *slots* da escala em que cada unidade poderia ser alocada. Este algoritmo apresenta alta complexidade temporal que pode ser representada na Figura 4:

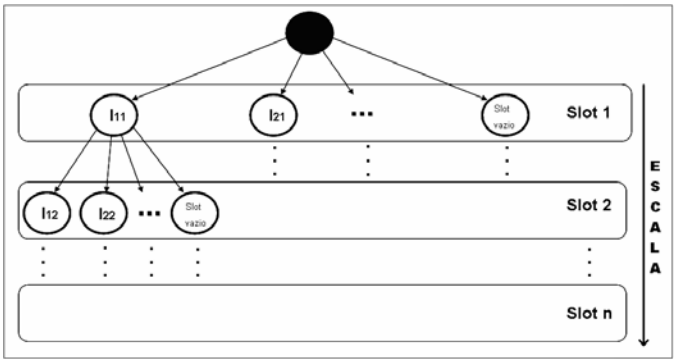

Figura 4. Representação do espaço de busca de um algoritmo força-bruta.

A complexidade do tempo de execução deste algoritmo é igual a $(I+1)^n$ combinações (todos os caminhos possíveis entre as folhas da árvore e o nodo raiz), onde I representa um número médio de instâncias que podem ser alocadas em cada um dos *slots* disponíveis na escala e n é a quantidade de unidades de execução que devem ser alocadas pelas instâncias.

Para tornar o algoritmo de escalonamento viável principalmente quanto ao tempo de execução é necessário realizar podas na árvore de busca evitando-se que escolhas erradas sejam feitas durante a busca. A complexidade espacial da árvore de busca pode ser reduzida se as unidades de execução das instâncias que podem ser alocadas no próximo *slot* da escala são definidas dinamicamente de acordo com as instâncias que estão em execução no momento, ou seja, o próximo nível da árvore de busca é criado considerando o nós percorridos até o momento eliminando combinações que não atendam as condições definidas a seguir:

- Relações de precedência e exclusão: Dentre as possíveis instâncias que podem ser alocadas no próximo *slot* de tempo da escala, aquelas que são precedidas ou exclusas por alguma das instâncias que está em execução não podem ser alocadas no próximo *slot* sem violar alguma restrição. Assim, este(s) nodo(s) são eliminados do próximo nível da árvore.
- Tarefas não-preemptivas: Se a instância que foi alocada no último *slot* da escala é não preemptiva e ainda possui unidades de execução a serem alocados, podemos afirmar que a única possibilidade de alocação do próximo *slot* seria outra unidade de execução da própria instância, visto que ela não pode ser preemptada.

Com a aplicação destas condições na alocação de cada *slot* de tempo da escala, o tamanho da árvore de busca é reduzido pela eliminação de algumas combinações que violam restrições do sistema.

A complexidade do tempo de busca ainda pode ser reduzida. Para atingir este objetivo foi adotada a seguinte estratégia: durante a geração da escala, na alocação de cada um dos *slots* de tempo da escala, são realizados cálculos para verificar se é possível que todas as instâncias que faltam executar respeitarão todas as suas restrições. Caso este cálculo indique que alguma restrição será violada, uma unidade de execução de outra instância deve ser alocada no *slot* de tempo atual da escala. Com isso, escolhas incorretas não são tomadas evitando-se que novos níveis da árvore sejam percorridos sem que exista a possibilidade de gerar uma escala viável.

Mais detalhadamente, este cálculo é realizado com segue:

- Para cada instância *I* que não executou completamente:
  - Seja *X* a soma dos tempos de execução restante de todas as instâncias de *deadline menor ou igual* ao deadline da instância *I*.
  - Seja *Y* a soma dos tempos de execução restante de todas as instâncias que estão em execução, *excluem* a instância *I* e que possuem *deadline maior* que o deadline da instância *I*.
  - Seja *Z* a soma dos tempos de execução restante de todas as instâncias que *precedem* a instância *I* e que possuem *deadline maior* que o deadline da instância *I*.
  - *X*, *Y* e *Z* correspondem à soma das unidades de execução das instâncias que devem executar até o tempo de deadline de *I* devido às restrições de deadline, exclusão e precedência, respectivamente.
  - A seguinte condição deve ser satisfeita: A quantidade das unidades de execução que devem ser alocadas no intervalo entre o *slot* de tempo atual e o deadline de *I* deve ser menor ou igual à quantidade de *slots* de tempo disponíveis (vide Figura 5a).

$$X+Y+Z \leq \text{Deadline}(I) - \text{SlotDeTempoAtual} \quad \text{(C1)}$$

Quando esta condição é verdadeira garante a existência de pelo menos uma escala que atenda todas as restrições temporais do sistema. Este cálculo ainda não considera o custo associado ao tempo gasto pelo despachante para realizar as trocas de contexto ocorridas no processador. Para considerarmos os custos de trocas de contexto na condição (C1) devemos acrescentar um novo termo *W*, onde *W* corresponde ao custo associado à ação de restauração do contexto de cada uma das instâncias que devem executar entre o slot de tempo atual e o tempo de deadline da tarefa I. Assim, a nova condição (Figura 5b) fica:

$$X+Y+Z+W \leq \text{Deadline}(I) - \text{SlotDeTempoAtual} \quad \text{(C2)}$$

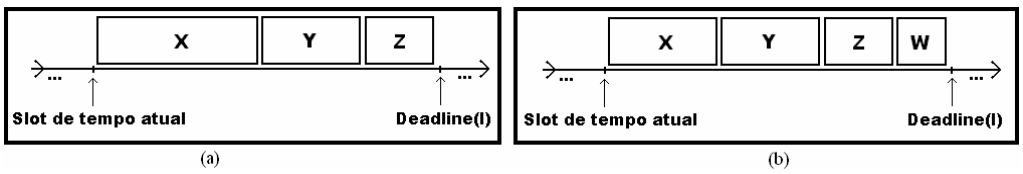

Figura 5. (a) Condição de previsão do atendimento do deadline da instância I. (b) Condição considerando o custo do despachante na previsão.

Podemos notar que o valor de *W* é baseado no menor tempo possível gasto pelo despachante, entretanto, se houver uma preempção entre tarefas na faixa de tempo considerada, o custo do despachante aumenta devido às ações adicionais de salvamento e restauração de contexto, portanto, esta previsão não garante que existe uma escala viável se a condição for satisfeita, entretanto, se existe uma escala viável, a condição é verdadeira.

6. Interface Gráfica

Visando oferecer aos usuários maior facilidade de utilização do algoritmo proposto foi desenvolvida uma interface gráfica em Java que se apresenta mais intuitiva para declaração do conjunto de tarefas do que um arquivo texto no formato especificado anteriormente (Figura 6a). Outra vantagem da interface gráfica consiste na apresentação da escala gerada pela ferramenta na forma de Diagrama de Gantt que proporciona ao usuário uma representação mais natural da escala gerada (Vide a Figura 6b).

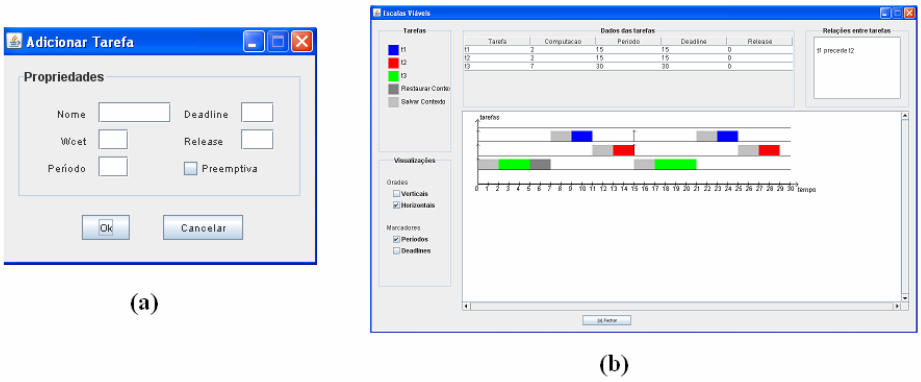

Figura 6. (a) Definição das propriedades das tarefas do sistema. (b) Apresentação da escala gerada na forma de diagrama de Gantt.

Como o algoritmo de escalonamento foi implementado em Prolog/Sicstus, tornou-se necessária uma interface entre a ferramenta desenvolvida em Prolog com a interface gráfica desenvolvida em Java. Para isto utilizamos uma biblioteca disponibilizada para o SICStus Prolog chamada prologbeans que possui predicados e métodos implementados em Prolog e Java, respectivamente, que se comunicam durante a execução dos dois processos como mostrado na Figura 7. Neste esquema, o programa escrito em Prolog funciona como um servidor de consultas, enquanto a aplicação Java deve solicitar a prova dos predicados implementados no servidor. Na realização de qualquer consulta, o servidor Prolog busca provar a regra solicitada na aplicação Java e, ao final, responde se o predicado é válido retornando inclusive valores de argumentos não-instanciados passados na consulta.

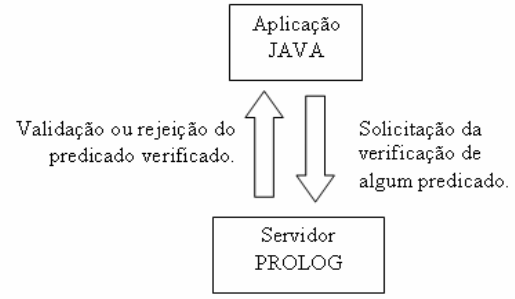

Figura 7. Interface disponível pela biblioteca prologbeans entre aplicações desenvolvidas em Java e SICStus Prolog.

7. Resultados obtidos

A ferramenta desenvolvida neste trabalho foi submetida a experimentos com o objetivo de analisar a viabilidade de utilização da ferramenta em tempo de projeto para geração de escalas para conjuntos de tarefas compatíveis com o modelo especificado na Seção 4. Os experimentos foram executados em uma máquina com processador Pentium-M de 1.7 Ghz com 2 MBytes de Memória Cache e 512 MBytes de Memória RAM. A distribuição do SICStus Prolog foi executada no sistema operacional Windows XP.

7.1. Conjunto de tarefas 01

Tabela 1. Representação do conjunto de tarefas 01

Tarefas	WCET	Deadline	Período	Release	Preemptiva?
Temp-sensor-start	1	1500	10000	0	sim
Temp-sensor-handler	1	1500	10000	11	sim
PWM	8	1500	10000	0	sim
Pulse-generator	4	50	50	0	sim
Temp-adjust-part1	1	5000	10000	0	sim
Temp-adjust-part2	2	5000	10000	1501	sim
t1 precede t2, t2 precede t3, t5 precede t6					
Restaurar Contexto = 10 e Salvar Contexto = 10					

Este conjunto de tarefas representa uma aplicação real responsável pela manutenção da temperatura da água em um valor constante. Podemos perceber que os tempos de execução das tarefas são mais curtas que o tempo gasto pelo despachante para efetuar a troca de contexto no processador. Este tipo de sistema sugere que o custo associado à execução do despachante não deve ser negligenciado durante a geração da escala, pois quando os tempos de WCET das tarefas do sistema são curtos tornam a execução do despachante relevante para que a escala gerada em tempo de projeto mantenha-se consistente durante a execução do sistema sem violação das restrições temporais.

7.2. Conjunto de tarefas 02

Tabela 2. Representação do conjunto de tarefas 02

Tarefas	WCET	Deadline	Período	Release	Preemptiva?
t1	30	161	161	0	Sim
t2	30	51	161	11	Sim
t3	30	90	161	60	Sim
t4	10	100	161	41	Sim
t5	50	140	161	90	Sim
t2 precede t4, t2 exclui t1 e t1 exclui t4					
Restaurar Contexto = 0 e Salvar Contexto = 0					

Este conjunto de tarefas foi idealizado sob medida por [Xu and Parnas 1993] de modo que a escala viável exige que os 11 primeiros *slots* de tempo permaneçam ociosos devido às relações entre t1, t2 e t4. A abordagem de escalonamento *offline* possibilita

que o algoritmo de escalonamento tenha uma visão do todo maior permitindo, por exemplo, que sejam feitas previsões das possíveis conseqüências da alocação de cada *slot* de tempo. Ao contrário, abordagens *online* tendem a gerar escalas considerando apenas dados atuais do sistema impossibilitando escalas para modelos de tarefas semelhantes a esses sejam encontradas.

7.3. Conjunto de tarefas 03

Tabela 3. Representação do conjunto de tarefas 03

Tarefas	WCET	Deadline	Periodo	Release	Preemptiva?
Vehicle Braking	3	11	30	0	Sim
Hazard Response	23	51	150	0	Sim
Sensor Data Fusion	10	80	500	0	Sim
Steering Control	4	20	20	0	Sim
Steering Set Point	3	11	50	0	Sim
Velocity Control	4	20	20	0	Sim
Velocity Set Point	3	11	50	0	Sim
System Management	5	50	100	0	Sim
CPU status	2	100	500	0	Sim
Electrical System Status	2	100	500	0	Sim
Power Train Status	2	100	500	0	Sim
T4 precede t6, t5 precede t7, t2 exclui t3 e t3 exclui t2					
Restaurar Contexto = 0 e Salvar Contexto = 0					

Este conjunto de tarefas representa uma aplicação denominada UGV (*Unmanned Ground Vehicle*) que possui 297 instâncias que devem ser escalonadas em 1500 slots de tempo. Devido ao grande número de instâncias e ao grande número de slots de tempo, este conjunto de tarefas é propicio para mostrar se o algoritmo de escalonamento é eficiente quanto ao tempo de execução.

Ao executarmos a ferramenta para este conjunto, em 3 segundos foi identificado que **não existe escala viável** para este conjunto de tarefas. Este tempo de resposta foi alcançado devido à previsão de deadlines que é realizada na alocação de cada *slot*, portanto, conjuntos de tarefas que não são escalonáveis são identificados na tentativa de alocação dos primeiros *slots* da escala. Ao diminuirmos a carga deste conjunto de tarefas de modo que seja possível gerar alguma escala viável, o tempo de execução para encontrar um escala viável foi de aproximadamente 4 minutos.

8. Conclusões

O projeto de sistemas embarcados de tempo real pode ser simplificado com o uso de ferramentas que sejam compatíveis com modelos de tarefas um pouco mais complexos e que considerem fatores mais próximos da realidade encontrada em aplicações reais. Isto reduz significativamente o custo associado ao tempo de projeto do sistema que utilizam a abordagem de escalonamento *offline*. Em sistemas críticos de tempo real, o funcionamento do sistema deve ser previsível para evitar que ocorram tragédias devido à alguma falha do sistema. Esta previsibilidade pode ser obtida na geração da escala em tempo de projeto quando o sistema possui tarefas fixas cujos tempos de execução no pior caso são conhecidos previamente. Para que a escala gerada em tempo de projeto

funcione corretamente durante a execução do sistema, o tempo gasto pelo despachante nas trocas de contexto ocorridas no processador não podem ser negligenciadas, pois existem aplicações reais em que o despachante consome mais tempo que as próprias tarefas do sistema. Mesmo em uma abordagem *offline* deve ser verificada a eficiência do algoritmo de escalonamento para conjuntos de tarefas maiores com o objetivo de verificar a viabilidade do tempo de execução. Neste trabalho foram realizadas podas durante a busca que viabilizaram a execução do algoritmo de escalonamento. As previsões de deadlines realizadas direcionam o caminho da busca por uma escala viável melhorando significativamente o desempenho do algoritmo, mesmo exigindo um tempo maior para realização do cálculo proposto.

Aplicações mais complexas podem tornar-se compatíveis com esta ferramenta se novos tipos de relacionamento entre tarefas forem incorporados ao modelo de tarefas como, por exemplo, as relações descritas em [Ekelin and Jonsson 1999]. Além disso, as escalas geradas podem ser otimizadas de acordo com alguma métrica de avaliação como, por exemplo, a quantidade de trocas de contexto ocorridas no processador. Entretanto, devido à complexidade do problema de escalonamento, heurísticas deverão der adotadas durante a busca da escala ótima para tornar viável a execução do algoritmo de escalonamento. Uma possível solução seria adaptar o algoritmo Minimax cuja função de avaliação seria uma análise das escalas de acordo com a métrica adotada.

Referências

Baker, T.P. and Shaw, A. (1988). The Cyclic Executive Model and Ada. In *Proceedings of the 9th IEEE Real-Time Systems Symposium*, pp. 120-129.

Ekelin, C. and Jonsson, J. (1999). Real-Time System Constraints: Where do They Come From and Where do They Go?. In *Proceedings of the Int'l Workshop on Real-Time Constraints*, Alexandria, Virginia, USA, pp. 53-57.

Ekelin, C. and Jonsson, J. (2001). A CLP Framework for Allocation and Scheduling in Embedded Real-Time Systems, *TR-01-10*, 2001.

Schild, K. and Wurtz, J. (1998). Off-line scheduling of a real-time system. In K. M. George.

Xu, J and Parnas, D. On satisfying timing constraints in hard real-time systems. *IEEE Trans. Soft. Engineering*, 19(1):70–84, January 1993.